Listing of Claims:

1 (Canceled)

2 (Previously Presented) The method of claim 31 further including:

(f) including a second set of methods in the adapter API for the client applications that provides consumption logic and methods for automatically exporting data defined in a Web-based schema registry, from the database into the client applications using Web services.

3 (Previously Presented) The method of claim 2 further including: registering the respective data sources with a schema registry in order to create the schema definition and the document type definition (DTD).

4 (Canceled).

5 (Canceled).

6 (Previously Presented) The method of claim 3 wherein the adapter API includes an XML API comprising the first set of methods and the second set of methods, wherein the first set of methods comprises a Writer API, and the second set of methods comprises a Reader API.

7 (Previously Presented) The method of claim 6 wherein the client applications are modified with generator logic that makes calls to methods comprising the adapter API, wherein once called, the Writer API converts data into the XML format in memory and saves the XML format data in the XML files, which are then transported to the server.

8 (Canceled).

9 (Previously Presented) The method of claim 7 wherein the adapter further includes verification code that verifies the generated XML data against the DTD defined in schema registry.

10 (Canceled).

11 (Canceled)

12 (Canceled)

13 (Canceled)

14 (Previously Presented) The computer-readable medium of claim 32 further including instruction of:

(c) including a second set of methods in the adapter API for the client applications that provides consumption logic and methods for automatically exporting data defined in a Web-based schema registry, from the database into the client applications using Web services.

15 (Previously Presented) The computer-readable medium of claim 14 further including the instruction of: registering the respective data sources with a schema registry in order to create the schema definition and the document type definition (DTD).

16 (Canceled).

17 (Canceled).

18 (Previously Presented) The computer-readable medium of claim 15 wherein the adapter API includes an XML API comprising the first set of methods and the second set of

methods, wherein the first set of methods comprises a Writer API, and the second set of methods comprises a Reader API.

19 (Previously Presented) The computer-readable medium of claim 18 wherein the client applications are modified with generator logic that makes calls to methods comprising the adapter API, wherein once called, the Writer API converts data into the XML format in memory and saves the XML format data in the XML files, which are transported to the server.

20 (Canceled).

21 (Previously Presented) The computer-readable medium of claim 19 wherein the adapter further includes verification code that verifies the generated XML data against the DTD defined in schema registry.

22 (Canceled).

23 (Canceled)

24 (Canceled)

25 (Canceled)

26 (Previously Presented) The system of claim 33 wherein the adapter API further includes a second set of methods for the client applications that that provides consumption logic and methods for automatically exporting data defined in the schema registry, from the database into the client applications using Web services.

27 (Previously Presented) The system of claim 26 wherein the adapter API includes an XML API comprising the first set of methods and the second set of methods, wherein the

first set of methods comprises a Writer API and the second set of methods comprises a Reader API.

28 (Original) The system of claim 27 wherein the server further includes a schema generator for generating the schema definition, a DTD generator for generating the DTD, and an adapter software kit that is downloaded from the server and used to incorporate the adapter API into the client applications.

29 (Canceled)

30 (Canceled)

31 (Previously Presented) A method for providing data integration and exchange between a plurality of client applications over a network, wherein each of the client applications access a respective data source, the method comprising:

(a) providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format, wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications;

(b) modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in XML files;

(c) submitting each of the XML files to an import repository at a server;

(d) validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the respective data sources; and

(e) parsing the validated XML files in the import repository and storing name/value pairs in a database at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications.

32 (Previously Presented) A computer-readable medium containing program instructions for providing data integration and exchange between a plurality of client applications over a network, wherein each of the client applications access a respective data source, the program instructions for:

(a) providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format, wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications;

(b) modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in an XML file;

(c) submitting each of the XML files from the client applications to an import repository at a server;

(d) validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the respective data sources; and

(e) parsing the validated XML files in the import repository and storing name/value pairs in a database at the server according to a hierarchy

specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications.

33 (Previously Presented) A data integration system, comprising:

a network;

a server coupled to the network, the server including a schema registry, an import repository, an XML loader, a database, and a published adapter API at each of the client applications that provides a first set of methods for translating data in respective data source into XML format; and

a plurality of client applications coupled to the network and in communication with the server, wherein each of the client applications access the respective data source, and wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications, and

wherein at least a portion of the client applications includes a corresponding schema definition and document type definition (DTD) registered with the schema registry, and the portion of the client applications includes generation logic for making calls to the first set of methods in the adapter API, such that data in the respective data sources are converted into XML format according to the corresponding schema definition and stored in XML files,

wherein each of the XML files is submitted to the import repository at the server, wherein each of the XML files in the import repository is validated against the corresponding DTD, and wherein the XML loader parses the validated XML files in the import reposition and stores name/value pairs in the database at the server according to a hierarchy specified by the corresponding DTD,

thereby standardizing the data from the data sources of the client applications.